NPS-CS-00-003

# NAVAL POSTGRADUATE SCHOOL
# Monterey, California

## NETWORK POLICY LANGUAGES: A SURVEY AND A NEW APPROACH

by

Gary N. Stone
Bert Lundy
Geoffrey Xie

August 2000

Approved for public release; distribution is unlimited.

Prepared for: DARPA / ITO
3701 Fairfax Drive
Arlington, VA 22203-1714

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

**1. AGENCY USE ONLY** ***(Leave blank)***

**2. REPORT DATE**
August 2000

**3. REPORT TYPE AND DATES COVERED**
Technical Report

**4. TITLE AND SUBTITLE**
NETWORK POLICY LANGUAGES: A SURVEY AND A NEW APPROACH

**5. FUNDING NUMBERS**
Order # G417

**6. AUTHOR(S)**
Stone, Gary N., Lundy, Bert, Xie, Geoffrey

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Naval Postgraduate School
Monterey, CA 93943-5000

**8. PERFORMING ORGANIZATION REPORT NUMBER**
NPS-CS-00-003

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
DARPA / ITO
3701 Fairfax Drive
Arlington, VA 22203-1714

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** ***(maximum 200 words)***

In this report a survey of current network policy languages is presented. Next, a summary of the techniques for detecting policy conflicts is given. Finally, a new language, Path-based Policy Language (PPL), which offers improvements to these is introduced.

Previous network policy languages vary from the very specific, using packet filters at the bit level, to the more abstract where concepts are represented, with implementation details left up to individual network devices. As background information a policy framework model and policy-based routing protocols is discussed. PPL's path-based approach for representing network policies is advantageous in that Quality of Service (QoS) and security policies can be associated with an explicit path through the network. This assignment of policies to network flows aids in new initiatives such as Integrated Services. The more stringent requirement of supporting path-based policies can be easily relaxed with the use of wild card characters to also support Differentiated Services and best-effort service, which is provided by the Internet today.

**14. SUBJECT TERMS**
Policy Language, Path-Based, Network Management, Conflict Detection, Conflict Prevention

**15. NUMBER OF PAGES**
48

**16. PRICE CODE**

**17. SECURITY CLASSIFICATION OF REPORT**
Unclassified

**18. SECURITY CLASSIFICATION OF THIS PAGE**
Unclassified

**19. SECURITY CLASSIFI- CATION OF ABSTRACT**
Unclassified

**20. LIMITATION OF ABSTRACT**
UL

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM Richard H. Wells, USNR
Superintendent

Richard Elster
Provost

This research was supported in part by DARPA under the Next Generation Internet Program (Order # G417), a grant from NASA Ames Research Center, and a fellowship from SPAWAR San Diego

This report was prepared by:

Gary N. Stone

GARY N. STONE
Ph.D. Candidate

BERT LUNDY
Associate Professor of Computer Science

GEOFFREY XIE
Assistant Professor of Computer Science

Reviewed by:

DAN BOGER
Chairman Department of Computer Science

Released by:

D. W. NETZER
Associate Provost and Dean of Research

## ABSTRACT

In this report a survey of current network policy languages is presented. Next, a summary of the techniques for detecting policy conflicts is given. Finally, a new language, Path-based Policy Language (PPL), which offers improvements to these is introduced.

Previous network policy languages vary from the very specific, using packet filters at the bit level, to the more abstract where concepts are represented, with implementation details left up to individual network devices. As background information a policy framework model and policy-based routing protocols is discussed. PPL's path-based approach for representing network policies is advantageous in that Quality of Service (QoS) and security policies can be associated with an explicit path through the network. This assignment of policies to network flows aids in new initiatives such as Integrated Services. The more stringent requirement of supporting path-based policies can be easily relaxed with the use of wild card characters to also support Differentiated Services and best-effort service, which is provided by the Internet today.

# TABLE OF CONTENTS

# ACKNOWLEDGEMENTS

This research was supported in part by DARPA under the Next Generation Internet Program (Order # G417), a grant from NASA Ames Research Center, and a fellowship from SPAWAR San Diego. The authors would also like to thank Prof. Bret Michael for his review of this report and his insightful comments.

# I. INTRODUCTION

Millions of dollars are lost, a company folds, and thousands of employees are let go, throwing a community into economic chaos. These events could result from discontinued funding from Silicon Valley investors who became aware of continued reports of poor QoS, security problems, and the inability for clients to access the company's web sites. This ingenious company was the first to integrate voice, video, and data all on the same network based on a company's policies. The company's untimely demise was the result of *conflicting network policies* that were disseminated automatically throughout the network's policy servers causing erratic network performance. This scenario, although extreme, shows the importance of being able to represent the network policy goals of a company while simultaneously verifying that those goals do not conflict with each other.

To understand how policy can play a role in managing a network, policy must be defined and applied to communication networks. The Internet Engineering Task Force (IETF) has proposed an Internet-draft of terminology for describing network policy [34] and provides many of the definitions used throughout this paper.

A policy is formally defined "as an aggregation of policy rules. Each policy rule is comprised of a set of conditions and a corresponding set of actions. The conditions define when the policy rule is applicable. Once a policy rule is so activated, one or more actions contained by that policy rule may then be executed. These actions are associated with either meeting or not meeting the set of conditions specified in the policy rule" [34]. In other words, a policy specifies what action(s) must be taken when a set of associated conditions are met.

A simple view of policy in regards to networks is that policy constrains communication. Specifically, network policy defines the relationship between clients using network resources and those network elements that provide those resources. A *client* in this case refers to users as well as applications and services.

Network policy allows administrators to manage network elements to provide service to a set of clients. If every system were permitted to communicate with all other systems without restriction, then there would be no need for network policies. Increasingly, networks that once only supported best-effort traffic are now integrating voice and data as well. Without a means for network managers to control the use of the network, mission-critical applications and general network performance is going to suffer and there will be little hope of supporting future real-time applications.

Network policies are grouped into three general areas:

1. how the policy is used
2. how the policy is triggered
3. what level the policy is applied

A *usage policy* describes *what* services will be used to maintain the current state of the network or to transition to a new state. Services which may be available in the network are differentiated service classes, virtual private networks, encryption capability, etc. A usage policy also describes *how* those services will be used. For example the ability to differentiate the handling of separate flows of traffic based on the service class they reside in, or which virtual channel they belong to, describes how a service is used.

Policies can be *triggered* in two ways, either statically or dynamically. "Static policies apply a fixed set of actions in a pre-determined way according to a set of pre-defined parameters that determine how the policy is used"[34]. Examples of static policies are: transit traffic in not permitted during normal working hours; Internet radio is only permitted after 4:00 PM; for security reasons certain network addresses are denied access to network resources.

Dynamic policies are only enforced when needed, and are based on changing conditions of the network such as congestion, packet loss, or the loss of a network router. To support the dynamic and sometimes unexpected nature of the network, actions can be triggered when an event causes a policy condition to be met. Examples of dynamic polices are: when the network gets congested streaming video traffic is disallowed; or when a particular service class of user is utilizing the network, lower best-effort traffic to only 25% of link capacity.

Lastly the *level of the policy* is applied as a category. These policies are differentiated by their granularity, such as the application level, user level, class level, or service level. For example a mission critical application may be given priority over all other network traffic, or all users in the silver class (differentiated service) have priority over the bronze class but must succumb to the gold class.

In section VII of this paper the authors introduce a policy language that is based on path. *Path-based policy* is defined to be a policy were all attributes associated with the policy, which include the service type of the traffic, conditions used to trigger the policy, and the actions executed when the policy is triggered, are all bound to a predefined path. Using path as the fundamental building block of a policy statement provides great control and flexibility. The ability to specify an explicit path, which represents each node from source to destination, enables us to create virtual channels where resources are reserved to support real time applications. These paths can either be specified by a user or by a network administrator. If a path had the restriction of *always* including each node in the path, then the number of unique paths needed to support a network could soon become over whelming. This is why a path may include wild card characters, and thus adds great flexibility to the way policies are specified. The use of a wild card character allows for path aggregation which greatly reduces the number of paths that have to be specified, and at the extreme one path statement can specify all possible paths under an administrator's control.

*Policy-based networking* - the ability to control a networking environment by specifying and enforcing policies - is gaining increased interest among the network community. Policy-based networking helps manage user and applications priority, quality of service and security rights, based on management policies. Because of an increasing industry trend to deploy business applications over the network and the convergence of voice, video and data applications on the same network, major network vendors such as Cisco, Nortel, and Lucent Technologies are developing products to support network management. These products allow network managers to create and implement policies that can prioritize the use of network resources by different network applications so that bandwidth will be guaranteed to the most business-critical applications during times of network congestion. For example, a company which offers IP telephony – which has strict timing requirements – must not permit a large data file transfer to interfere. Network management also provides the ability to restrict the use of network segments by denying access of unwanted and perhaps malicious traffic. The ability to create and enforce network policies adds intelligence to a network that was previously based only on best-effort packet traffic. Rather than adding more bandwidth, which is expensive and time consuming, to solve existing network congestion, companies can use network policies to allow for important applications and user groups to receive network priority over secondary network users.

Many aspects of policy-based networking are being addressed such as policy storage structures, policy servers, and protocols to deliver translated policies to enforcement points. One aspect of policy-based networking that does *not* seem to be receiving much attention is the verification of policies that are going to be applied to the network. Consistent enforcement of network policies, often specified by different people at different times, is impossible if those policies conflict with each other. Thus, a method is needed to detect and deal with conflicting policies before they are distributed throughout the network to the policy enforcement points.

## A. IETF POLICY FRAMEWORK CORE INFORMATION MODEL

With the emergence of service models such as Differentiated Services (DS) [27,41], Integrated Services (IS) [27,41], and Multiprotocol Label Switching (MPLS) [41], the IETF has published a working draft for terminology to describe network policies and services [36]. This draft attempts to develop a scalable framework for policy administration and distribution of network policies across multiple devices and multiple vendors. A key to this framework is a common language to represent and provide a consistent implementation of policy.

An underlying assumption of this draft is that policies are stored in a centralized repository. The policy repository is one of three important entities of the model. The other entities are the Policy Enforcement Points (PEP) and Policy Decision Point (PDP).

The PEP is a component of a network node (e.g., a router, switch, or hub) where the policy decisions are actually enforced. When the PEP requires a policy decision about a new flow of traffic, or authentication for example, the PEP will send a request to a PDP.

The PDP is the entity in the network where policy decisions are made. This PDP, which may reside on a remote server, will make policy decisions using information retrieved from policy repositories.

Communication is needed to and from the policy repository as well as between the PDP and the PEP. In many proposals the policy repository is a directory and therefore the appropriate access protocol would be the Lightweight Directory Access Protocol (LDAP). Examples of a policy protocol, which is used to request and reply to policy decisions, could be the Common Open Policy Service protocol (COPS) [4] and the Simple Network Management Protocol (SNMP) [6].

Since the PEPs can potentially be from multiple vendors, a common policy language is needed to support the dissemination of policy information to these devices. In the Policy Framework Core Information Model [36], policy is defined as an aggregation of policy rules. Each of these policy rules is composed of a set of conditions and a set of actions to perform if the conditions are met. The general form of these conditional statements is shown below.

**IF** <condition 1> **AND** <condition 2> ... **AND** <condition N>
**THEN** <action 1> ... **AND** <action N>

The policy representation includes a means to prioritize and order both the conditional statements as well as the policy actions. This is crucial when multiple policies exist and these policies conflict. A conflict occurs when the conditions of at least two policies are simultaneously satisfied, but the actions of at least one of the policies can not be simultaneously executed. For example, a router may have two access control rules where their conditions are simultaneously satisfied, but one contains that action deny, the other permit. For example:

```
access-list 1 permit 131.1.30.0  0.0.0.255
access-list 1 deny   131.1.0.0   0.0.255.255
```

The first permits traffic with IP addresses beginning with 131.1.30 to pass. The second rule conflicts with the previous one by denying traffic with any IP address beginning with 131.1. The first rule in an access list that satisfies the conditional requirement is executed. This procedure resolves conflicts but puts the onus on the operator to enter the rules in the correct order.

## B. REPORT ORGANIZATION

The rest of this report is organized into seven sections. Section II reviews policy-based routing protocols. A lot of early work on the use of polices in networks occurred in the context of these protocols. Section III discusses languages that are used to represent network policies. These languages are at a more abstract level and can be used to describe policies without low-level details. An abstract language is beneficial when multiple devices and vendors are involved. Section IV discusses languages used to describe network traffic at a low level such as the Protocol Data Unit (PDU) level. These languages are more adept at defining patterns for the selection of network traffic in the conditional section of a policy.

Section V summarizes the existing network policy languages. Section VI contains a review of research efforts that involved using formal logic to determine consistency between policies. This formal logic section provides background information for features introduced with our new language. Section VII introduces our language called the Path-based Policy Language (PPL). Our goal is to represent network policies at an abstract level in order to support heterogeneous networks, while also providing the translation of those policies into formal logic. Having policies represented in logic will provide the ability for theorem provers to detect conflicts. Section VIII summarizes the report.

# II. POLICY-BASED ROUTING PROTOCOLS

In this section three policy-based routing protocols are reviewed. They provided a lot of early work on the use of policies in networks. All of these protocols enable policies to be enforced based on the elements of an explicit path through the network.

## A. BORDER GATEWAY PROTOCOL

Lougheed and Rekhter define an inter-autonomous routing protocol, Border Gateway Protocol (BGP) [14, 15, 16], where routers share reachability information by passing Autonomous System [1](AS) information between neighbors. This exchange of routing information contains full AS paths that the traffic will transit to reach a distant network. Path information is not only useful in removing loops in the network, but also allows policy decisions to be made at the AS level. Policy enforcement is not part of the protocol itself but instead is manually configured at each BGP router.

Policy decisions made by BGP [17, 10] are based on configuration information manually configured into each router by an AS administrator. The enforcement of policies is accomplished in two ways. The first is by specifying the procedure by the AS router itself to select the best paths, and the second is by controlling the redistribution of routing information to neighboring ASs.

Policy decisions can be based on various preferences and constraints. Since the complete AS path is advertised to neighboring routers, particular paths can be rejected based on an AS that is contained in the path. The reasons a particular path are rejected vary. For example a particular AS whose control is under that of a major competitor may want to be avoided, causing one or more paths that include this AS to be eliminated from consideration. Performance information can also be used to eliminate paths from consideration. If an AS has access to metrics related to performance such as link speed, delay, or capacity, then these measurements can be used to rate multiple paths for selection.

BGP allowing an AS to control redistribution of routing information is the means by which an AS can enforce policies on others. For example, if an AS does not want to be used for transit traffic, then it does so by not advertising routes to networks other than those directly connected to it.

Fundamentally BGP is a distance vector protocol, but instead of maintaining just the cost to each destination, BGP keeps track of the exact path used. As mentioned earlier policies are not part of the BGP protocol itself and therefore each AS may have its own means for evaluating paths. Each router contains a module for examining paths to a given destination and scores them. This scoring mechanism, which may include local policy information, is then used to choose the best path to a destination.

BGP routers can only advertise paths that itself uses. This prevents an AS from sending datagrams to a distant network using one path, but advertising an alternative path for others to use. This "hop-by-hop" routing paradigm which is generally used by the current Internet prevents the support of source routing.

## B. INTER-DOMAIN ROUTING PROTOCOL

Kunzinger and Thomas describe the Inter-Domain Routing Protocol (IDRP) [11, 38], which is the International Standards Organization's (ISO) protocol for routing between Autonomous Systems. Just as in BGP, IDRP supports policy-based routing, but is not concerned with the implementation details of

[1] Autonomous System (AS), Administrative Region (AR) are a set of routers under a single technical administration, using one or more interior gateway protocols to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs[n3].

those policies. Policy-based routing can restrict access, and therefore enforce policy, by controlling the distribution of routing information to neighboring routers. This selective distribution of information can enable the AS to deny all transit traffic, or may deny access to only certain network paths.

The IDRP router accepts router information from neighboring routers, which express their views of the network, and uses this gathered information to construct it's own view of the network. The IDRP router at this point can use local policy information to select or reject routes accordingly. The IDRP router advertises it's view of the network with internal gateway protocols such as Open Shortest Path First (OSPF) or Routing Information Protocol (RIP) so that all routers within the AS have a consistent view of the network.

Just as autonomous systems were used to refer to an entire set of IP networks, IDRP supports a concept call routing confederations. A routing confederation is a grouping of autonomous systems to make managing the Internet more manageable. As the Internet has grown, the number of autonomous systems has also grown making it's management less efficient. These routing confederations are quite flexible in that they can be subsets of each other, and can even overlap each other.

IDRP uses path vector routing to propagate routing information. Path vector routing, like BGP, explicitly lists the entire path to each destination. This concept can alleviate network loops as well as enforce policy constraints based on the autonomous systems or confederations that comprise the path.

Another feature supported in IDRP is the ability to reduce the number of path vectors by using route aggregation. Route aggregation lets an IDRP routers combine multiple IP address prefixes, destinations, to create a single advertisement for them all. This feature greatly reduces the number of individual destinations a router must support as well as reducing the amount of data that has to be sent during the advertising phase.

## C. INTER-DOMAIN POLICY ROUTING (IDPR)

Steenstrup presents a set of protocols [32] and an architecture in [33] for Inter-Domain Policy Routing (IDPR). IDPR is a routing protocol that provides policy routing among Administrative Domains (ADs)[2]. The primary objective of IDPR is to provide traffic with routes that satisfy the users' service requirements while respecting the service providers' service restrictions [31]. *Source policies* represent the users' requirements and can consist of parameters, such as throughput, acceptable delay, cost of session, domains to avoid, etc. Service providers specify *transit policies* which specify offered services and the conditions of their use.

During route generation and selection, routes are filtered out which are not consistent with both the source and transit policies. Route generation is inherently complex and the most computationally intensive part of IDPR. The general policy route generation problem involves a combination of service constraints. For example, finding a route that generates minimum-delay and least-cost. Trying to calculate such a route is an NP-complete problem.

To transport data along a selected route, a hop-by-hop or source specific method can be used. With hop-by-hop forwarding, each router makes an independent forwarding decision based on its forwarding information database. If all the routers have consistent information then the result is the same as source specific. With source specific, the source domain dictates the data message forwarding decisions to the routing entities in each intermediate domain, which then forward data messages according to the source specification.

To reduce the size of the link-state database, IDRP supports the ability to group ADs into super domains. The existence of super domains imposes a domain hierarchy within the network. With a

[2] Administrative domain (AD) refers to any collection of contiguous networks, gateways, links, and hosts governed by a single administrative authority who selects the intra-domain routing procedures and addressing schemes, specifies service restrictions for transit traffic, and defines service requirements for locally generated traffic.

hierarchical approach only domain level information is needed to construct routes. This greatly reduces the information needed to be maintained by a route server. The size of the database will now depend on the number of domains and the policies associated with each.

A variant of Clark's policy term, section III(A), was chosen to represent policies in [33]. This variant allows for policies to be associated with a set of network elements that represents a path. A policy based on path is a great asset to policy-based routing protocols.

THIS PAGE INTENTIONALLY LEFT BLANK

# III. NETWORK POLICY LANGUAGES

In this section all the major policy languages are discussed. These languages are used to represent varying types of network policies such as routing, access, and QoS.

## A. CLARK'S POLICY TERM

Seeing the importance of using network resources differently and more efficiently, Clark proposed a template to represent network policies [8]. This template, called a *Policy Term,* was designed to enable a wide range of network policies to be represented. The work is based on the fundamental assumption that Internet resources are grouped into Administrative Regions (ARs). ARs resources included such items as networks, links, routers, and gateways. The format of a Policy Term is shown in figure 1.

The first two "elements" of the Policy Term represent the source and destination points respectively. Each of these two points consist of three parts which provide for a wide range of granularity while specifying the end points. To show the granularity available with this schema, here are some examples of source and destination points that could be represented with the diagram in figure 2. These source and destination points use the special characters "*" and "-". The "*" represents the wild-card match and the "-" is used to make sure the AR entry or AR exit fields match the source AR or destination AR respectively. These examples could be applied to AR 2.

((*Hs*, *ARs*, *ARent*), (*Hd*, *ARd*, *ARexit*), *UCI*, *Cg*)
where:
*Hs* is the source host address
*ARs* is the source AR
*ARent* is the entry AR (previous hop)
*Hd* is the destination host address
*ARd* is the destination AR
*ARexit* is the exit (last hop)
*UCI* is the User Class Id (e.g. Gold, Silver Bronze service levels)
*Cg* are any global conditions

**Figure 1. Policy Term. After [8]**

(*, *, *) (*,*,*)

No restrictions, allow all traffic flows to traverse without restriction.

**(*, 36, -) (*, 12,*)**

Allow all hosts directly attached to AR 36 to pass if their destination goes through AR 12 (e.g., host 131.120.1.13 may communicate with 216.34.20.1)

**(131.120.1.13, 36, -) (216.32.74.53, 2, - )**

The host with IP address 131.120.1.13 in AR 36 may communicate with the host with IP address 216.32.74.53 in the AR 2.

As the reader can see, the end points can be as explicit as specifying a host or generic enough to allow all Internet traffic. Although the use of these first two elements provides for a flexible way to permit traffic flow, it can become cumbersome at times. If, for example, the reader wanted to allow only traffic from universities to flow across an AR, it could be accomplished by creating many policy terms, one for each university. This list of policy terms could become quite large, so the third "element" of the policy term, UCI, can be used to make the implementation of this policy more manageable. A policy term that would only allow university traffic to flow across an AR could be represented like this:

**((*,*,*), (*,*,*), University, *)**

The end points are such that if the UCI element was not used, then all traffic would flow across the AR. Using the UCI as a filter, only traffic marked with a **University** tag would be permitted to pass.

The last element field of the policy term, Cg, is used for global conditions. Examples of information that might be held in this field are, "unauthenticated UCI", "no-per-packet charge", and "limited to n% of available bandwidth".

**((*,*,*),(*,*,*), University, {unauthenticated UCI})**

This would allow only traffic marked as **University** to flow through the AR. There is no need to verify that the packet traffic was really from a university host.

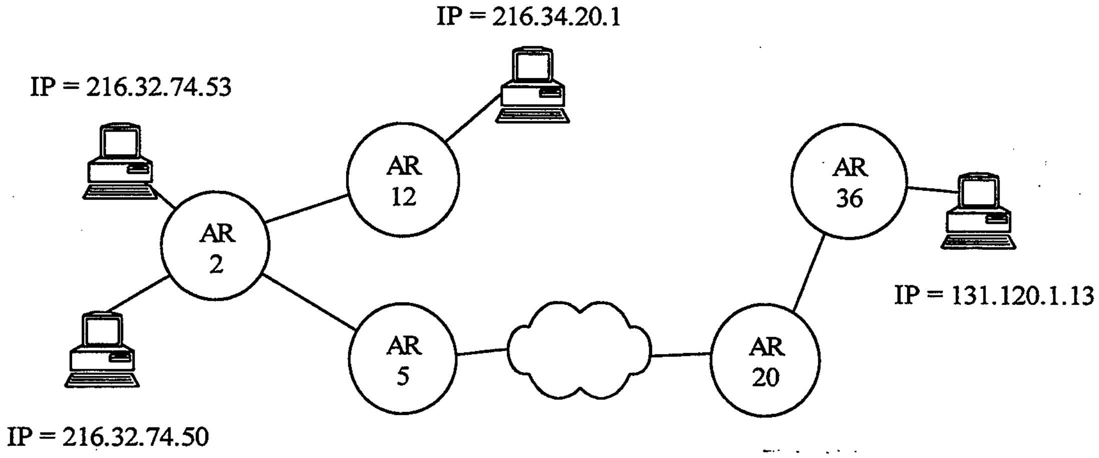


**Figure 2. Sample Network Diagram**

Although this was a good start for an abstract network policy representation, which is needed for heterogeneous environments, it has limitations. The first limitation is that there is no ability to represent explicit paths formed by a sequence of ARs as part of the term. Only single ARs and the wild card character "*" are allowed. Without this capability, support of Integrated Services requires several network policies distributed throughout the network to be combined for verification of a path. Consequently, there is no ability to exclude a set of ARs from a term to which a general policy is applied. Lastly, it may be desirable to represent a policy that is directional, so that a connection can be opened in one direction that has different conditions than the reverse flow. As defined in the language policy terms are bi-directional.

## B. POLICY FRAMEWORK DEFINITION LANGUAGE (PFDL)

Strassner and Schliemier define the language *PFDL* [35] that provides a mapping of network service requirements from a business specification to a vendor- and device- independent format. The benefits of such a language is that network policy can exist in a heterogeneous environment of devices that support policy enforcement.

With the development of standards to provide QoS, like integrated services with RSVP[3] and differentiated services, the IETF working group on Policy Management has proposed this language. The belief is that without a means for representing, administrating, and distributing consistent policy information, these QoS standards that classify and give preferential treatment to certain types of traffic flows will not see wide-scale deployment.

In this first release of the draft, the grammar was only available in Backus-Naur Form (BNF) and no explicit examples were presented. Attributes the authors believe should be supported by the language are discussed. As with many of these efforts to represent policy, the authors believe that having a language that will support multiple network devices and vendors is the key to successful policy deployment.

The design of PFDL is based on the Common Information Model (CIM) [36] being designed by the Distributed Management Task Force (DMTF). This model defines a hierarchy of object classes that can be used to represent policy information.

[3] Resource Reservation Protocol defines how applications can place reservations, and how they can relinquish those resources once their need ends.

The class and relationship hierarchy of the CIM model are used to help define the structure of the PFDL grammar, see figure 3. The basic premise is that a policy is an aggregation[4] of policy rules. A policy rule defines a sequence of actions to be initiated when a corresponding set of conditions is satisfied. Five classes defined to support the CIM are the *ComplexPolicy* class, *SimplePolicy* class, *PolicyRule* class, *PolicyCondition* class, and the *PolicyAction* class. Their relationship to each other is shown in figure 3.

A *PolicyRule* contains a set of *PolicyConditions* and a set of *PolicyActions*. When the set of *PolicyConditions* are meet, the set of *PolicyActions* will be executed.

A *PolicyConditionStatement* is composed of a category and value pair. These two components are specific to a particular knowledge domain, whether the domain be QoS, security, or any other domain. Providing conditions and actions for a given knowledge domain accommodates the interoperability requirement for the language. It will provide the means for multiple vendors to supply components to a general policy architecture.

A *PolicyAction* is a class in the PFDL model that consists of an action or a list of actions that will be executed when the conditions associated with a policy are evaluated to true. These actions can either be executed in a specific order, or any order which is the default. Along with the ordering of policy actions, the ability exists for the conditional execution of one or more actions based on the results of previous actions. The reader can see from figure 3 that the hierarchy of the PolicyActions class is similar to the *PolicyConditions* class.

With possibly hundreds, perhaps thousands of policies to be supported in a network, the ability to detect conflicting policies is crucial. The authors of PFDL are aware of the need to both detect as well as support facilities to resolve conflicts. This proposal groups policy conflicts into two different categories, intra-policy and inter-policy conflicts.

Intra-policy conflicts are caused when the conditions of at least two policies are simultaneous satisfied, but the execution of the actions of these policies cannot be executed at the same time. Inter-policy conflicts are described as two or more policies that, when applied to the network, result in conflicting configuration commands to be specified for one or more network devices. In this case, the conflict exists when the policy is applied to a specific network or device(s). An example given in the proposal is when two policies are executed such that the number of queues in one network device is such that it does not match the number of queues allocated in a second device supporting the same traffic flow.

Once conflicting policies are detected, they may be resolved in several different ways. The most obvious would be to modify the conditions or actions of the policies to remove the conflict. If this cannot be accomplished and the conflicting policies must exist in the system, there are three different ways to resolve them:

[4] An aggregation is a string form of an association. An aggregation is usually used to represent a "whole-part" relationship.

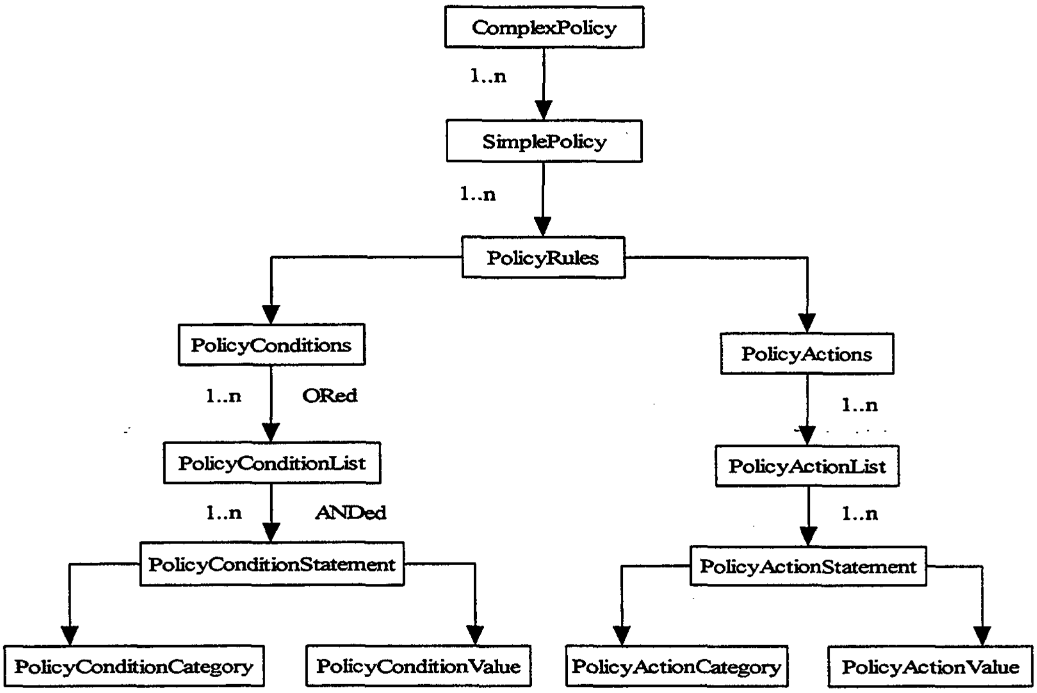


**Figure 3. PFDL Hierarchy. After [36]**

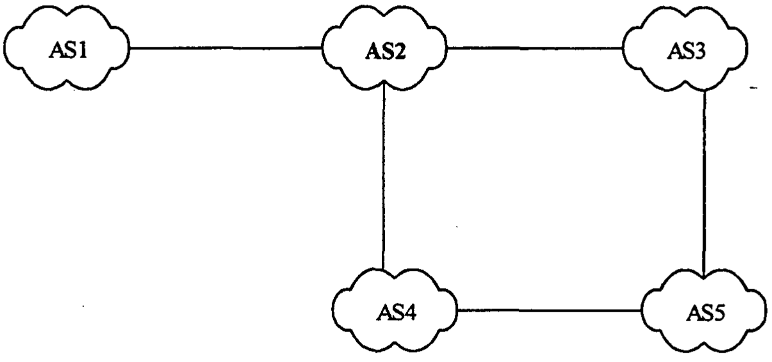
1) Resolve the conflict by only executing the first policy in the conflicting set.
2) Use a priority scheme where only the highest priority policy in a conflicting situation will be executed.
3) Use some type of metadata to determine which rule should be applied. The difference between this and straight priority is that priority is inherently linear, whereas metadata enables non-linear solutions, such as branching, to be used.

PFDL does not support path-based policies. A Path-based capability aids in initiatives such as Integrated Services [27, 41] and SAAM (Server and Agent based Active network Management) [42]. PFDL is a nice high-level framework, but lots of details need to be filled in.

## C. RPSL – ROUTING POLICY SPECIFICATION LANGUAGE

One of the activities of the Routing Policy System working group of the IETF is to develop a language for describing routing policy constraints. Alaettinolgu, Meyer et. al. provide a reference for the language [2] and a guide on how to use the language [20]. RPSL is a replacement for RIPE-81, the first language deployed in the Internet for specifying routing policies, and is the current Internet policy specification language. Specifying policies in RPSL allows a network operator to specify routing policies in the Internet Routing Registry (IRR), so that policies and announcements can be checked for consistency. The IRR stores the object-oriented policies of authorized organizations so that they can be queried by others using the whois[5] service. Each object which contributes to a policy stores pieces of information regarding the policy. Each object used to represent the policies contains attributes referred to as keys, that can either be mandatory or optional. RPSL is designed so that router configurations can be generated from the policies described with the language.

Figure 4 is an example from [20] that represents a common but perhaps simple policy. The *aut-num* represents the Autonomous System number, in this case AS2 represents autonomous system 2. The *as-name* and *descr* attributes are the Autonomous System's name and description, respectively. The most important attributes of this aut-num are the import and export policies. The import clause specifies the import policies, while the export clause specifies export policies.



```
aut-num:      AS2
as-name:      CAT-NET
descr:        Catatonic State University
import:       from AS1 accept ANY
import:       from AS3 accept <^AS3+$>
export:       to AS3 announce ANY
export:       to AS1 announce AS2 AS3
admin-c:      AO36-RIPE
tech-c:       CO19-RIPE
mnt-by:       OPS4-RIPE
changed:      orange@ripe.net
source:       RIPE
```

**Figure 4. RPSL diagram and policy example. From [20]**

In this example, the import policy of "from AS1 accept ANY" indicates that AS2 will accept any announcements that AS1 sends. The second import policy states that AS2 only accepts announcements from AS3 which originated in AS3 and have paths composed of only AS3's.

The export policy of "to AS3 announce ANY" indicates that any route that AS2 has in its routing table will be passed on to AS3. The second export will allow the announcements of all routes from AS2 or routes learned from AS3 to be sent to AS1.

The *admin-c* (administrative), *tech-c* (technical), *mnt-by* (maintained by), and *changed* (last changed by), are attributes that contain contact information. The values assigned to these attributes are

[5] WHOIS is used to look up records in a Whois database. Each record has a "handle", a unique identifier assigned to it by the Network Information Center (NIC). Each whois record will also have a name, a record type, and various other fields of information, all depending on the type of whois record

handles that uniquely identity the person responsible for the attribute. The *source* entry indicates that this object belongs to the RIPE[6] registry.

RPSL represents routing policies well, but was not intended for supporting policies regarding QoS or general access control mechanisms.

[6] The RIPE Network Coordination Centre acts as the Regional Internet Registry (RIR) for Europe and surrounding areas

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. TRAFFIC FLOW LANGUAGES

This section discusses languages that are used in the selection of network traffic in the conditional section of a policy. At a lower level than the languages in section III, these languages can be used for pattern matching in network devices.

## A. PAX PATTERN DESCRIPTION LANGUAGE (PDL)

Nossik, Welfeld, and Richardson describe PAX [25], a special purpose language used for defining pattern matching criteria in policy-based networking devices. PAX was intended primarily for data communications networks, but is also generic enough to be used for any kind of pattern recognition.

The language itself was designed to be much like that of the C programming language. Viewing the code of a PAX program, the reader will see features similar to C such as comments, preprocessing directives, source file inclusion, conditional compilation, import and export statements, and the use of defines and macros.

The basic concept in PAX is the pattern, with simple patterns being combined to form more complex patterns. The use of field concatenation, field combination, and the ability to name patterns leads to a flexible and powerful language for describing patterns in data communication.

Examples from [25] will provide the reader with a quick idea of the syntax and features of the language. Two built-in fields used in the following examples are BIT and UINT and are used to create the simplest of patterns.

*BIT 16* - matches any 16 bits in the input
*UINT 4* - matches any 4 bits and value of those bits are converted to an unsigned numeric field

Figure 5 illustrates a pattern to match IP version 4 headers of TCP/IP non-fragmented packets without IP options. This figure shows how simple patterns can be concatenated together to form more complex patterns. This pattern matches the input only when the 4 bit "version" element is equal to decimal 4, the next element "ihl" equals the decimal value 5, and the subsequent simple patterns are all successfully matched.

```
{
        version UINT 4 = = 4;               /* IP version 4 packet */
        ihl UINT 4 = = 5;                   /* length == 5 : no options */
        typeOfService UINT 8;
        totalLength UINT 16;
        identification UINT 16;
        flagReserved BIT 1 = = 0;
        flagDontFragment BIT 1;
        flagMoreFragments BIT 1 = = 0;      /* last fragments only */
        fragmentOffset UINT 13 = = 0;       /* first fragments only */
        timeToLive UINT 8;
        protocol BIT 8 = = 6;               /* Next protocol TCP */
        headerChecksum BIT 16;
        sourceAddress BIT 32;
        destinationAddress BIT 32;
}
```

**Figure 5. PAX pattern to match IPv4 header. From [20]**

Figure 6 represents two more features of the PAX language. The first being the ability to name the pattern for inclusion in other more complex patterns, in the case the name being IEEE_802_2_LLC. The second feature illustrated here is the use of a conditional field. Conditional fields are used to describe patterns with varying layouts depending on previous fields. In this case when the Controll field is equal to 0b11, the next 6 bits are used to create a field called ShortControl. When the value of Controll is not equal to 0b11, then the next 14 bits are used to create a field called LongControl.

```
PATTERN IEEE_802_2_LLC {
        DSAP BIT 8 <> 0xFF;        /* Destination SAP not broadcast */
        SSAP BIT 8 <> 0xFF;        /* Source SAP not broadcast */
        Controll BIT 2;
        LongControl BIT 14 WHEN Controll <> 0b11;
        ShortControl BIT 6 WHEN Controll == 0b11;
}
```

**Figure 6. PAX pattern with conditional field. From [20]**

## B. SIMPLE RULESET LANGUAGE (SRL)

Brownlee describes the Simple Ruleset Language (*SRL) [5]*, as a procedural language for creating rulesets for Realtime Traffic Flow Measurement (RTFM). These rulesets, which specify the flows to be measured and how much information should be collected for each, are downloaded to RTFM meters. The RTFM meters use a pattern matching engine to match the downloaded rulesets against attributes extracted from traffic flows to select which flows to monitor. The attributes applied to the traffic flows are specific to network traffic and map to such things as source and destination addresses, port numbers, etc. SRL is not restricted to just traffic metering, but can be useful in any application that involves selecting traffic flows from a stream of packets.

There are two goals of SRL rulesets, which are to identify network packets that are a part of the flow of interest, and then to take some action as a result of the match. The identification of packets is done using IF statements. Actions that are available include the ability to save flow identification attributes, and to keep statistical data about the attributes that are saved.

Figure 7 is an example that counts only TCP/IP packets were the destination port is telnet while saving the source and destination address pair for each packet.

```
#
# Classify IP port numbers
#
  define Ipv4 = 1;           # Address Family number from RFC 1700
  define telnet = 23;        # Well-known Port numbers from RFC 1700
  define tcp = 6;            # Protocol numbers from RCF 1700
#
  if SourcePeerType == Ipv4 save;
  else ignore;               # Not an Ipv4 packet
#
  if (SourceTransType == tcp && DestTransAddress == telnet)
      save, store FlowKind := 'T';
#
  save SourcePeerAddress /32;
  save DestPeerAddress /32;
  count
#
```

**Figure 7. SRL Ruleset to identify and count telnet packets. After [5]**

THIS PAGE INTENTIONALLY LEFT BLANK

# V. SUMMARY OF NETWORK POLICY LANGUAGES

Table 1 summarizes the languages from sections III and IV that are used to represent network policies or support mechanisms for enforcing policies. Policy-based routing protocols from section II are also represented in the table. The columns of this table represent criteria believed to be useful in comparing the various languages. The "Support Automated Conflict Detection" column refers to the ability to recognize different types of policy conflict as well as the ability to provide a flexible means to resolve conflicts. The column labeled "Suitable for Integrated Services" takes into account the ability to efficiently support Integrated Services. An entry with a "Medium" value signifies that the language can represent a path through the network, but that multiple polices have to be combined to do so. If a column had received a "High" value, then the policy language can represent a path in a direct and intuitive manner, and a policy can be applied directly to that path. The benefits of a "High" value are that policies that must be associated with all the nodes along a path can be represented with just one statement. This greatly reduces the number of policies statements that a domain must maintain. The "Suitable for Access Control" column refers to the ability to permit or deny access based on policy. The capacity to establish a path through a network and restrict access to that path, is of great importance from a security point of view. The column labeled "Target Architecture" refers to the storage location of the polices. A "Distributed" value means that policies are stored throughout the network, perhaps on individual devices. A centralized value refers to one, or just a few locations where all the policies are located. Having a "Centralized" location is beneficial when trying to detect conflicting policies. The last column, "Ease of Representing Network Policies", takes into account the ability of a user to intuitively represent a policy with the language. Targeting our language to the group of individuals responsible for representing policies defined with natural language and entering them into a central repository, the authors believe the more abstract and closer to natural language, the easier they will be understood. Although these individuals may be versed in formal logic representation, it is believed the majority will be more comfortable with an abstract rule-based language. The more abstract the language and closer to a natural language the higher the value in the column. The greater number of details that have to be specified, the lower the value assigned

| Language | Support Automated Conflict Detection | Suitable for Integrated Services | Suitable for Access Control | Target Architecture | Ease of Representing Network Policies |
|---|---|---|---|---|---|
| Policy Term | Low | Medium | High | Distributed | High |
| PFDL | Medium | Medium | High | Centralized | High |
| RPSL | Low | Low | High | Centralized | Medium |
| PAX | Low | Low | High | Distributed | Low |
| SRL | Low | Low | High | Distributed | Low |
| Policy-based Routing | — | High | High | Distributed | --- |

**Table 1 Summary of languages that represent or support network policies**

Although the languages represented in table 1 contain many features, none of these languages individually contain all the features provided with PPL. Two major features not adequately addressed by any of these languages are the ability to specify a complete path through a network, and the automatic detection of conflicting policies. The policy-based routing protocols, which are summarized in section II, are not concerned with the language used to represent network policies, but instead concentrate on supporting policy-based routing. As a result, columns that have no relevance to these protocols are filled with a "---".

# VI. WORK IN LOGIC REPRESENTATION OF POLICIES

This section discusses research using formal logic to represent and detect conflicting policies. The policies represented in this section are more general and involve the use of natural languages, which tend to be ambiguous, to represent policies ranging from resource management to security.

There has been a great deal of research on the topic of formal representation of policies. Much of this research has been in the area of representing security policies, and the more general problem of translating ambiguous natural language policies into some type of formal representation. Representing network policies with an unambiguous language is the key to detecting conflicts. A network policy language has to be flexible enough to represent a wide range of policies, at the same time being formal enough to support automatic translation to logic. Once the policies are in a logical representation, methods already developed from research in this area will provide a means for checking the consistency of multiple policies.

## A. ANALYZING CONSISTENCY OF SECURITY POLICIES

In Analyzing Consistency of Security Policies[7], the development of a methodology for reasoning about properties of security policies is discussed. Chovly and Cuppens view a security policy as a specific case of regulation, where a regulation defines what actions an agent is permitted, obliged or forbidden to perform. With this methodology a system is made up of agents which can perform some actions on some objects. In analyzing the consistency of security policies, focus is put on the ability to perform consistency checks (e.g., check for conflicting situations) on the system, and to have the ability to query a regulation to know which norms apply in a given situation.

Formal logic is used to create an unambiguous representation of security polices. According to Chovly and Cuppens [7] the advantage of a representation based on formal logic is the ability to precisely define the axioms[7] to reason about a regulation. With policies defined by axioms, tools can now be developed to check the system regulation for consistency.

Rather than associating norms (i.e., permissions, obligations, and prohibitions) with individuals, roles are created with these attributes and then individuals are associated with these roles. The individual inherits the norms associated with a role when the individual is playing that role. A conflict can only exist when an individual is playing different roles at the same time, because of an assumption in their research that norms within a role are conflict free.

To resolve conflicts when an individual is playing multiple roles, an ordering is applied when roles are merged. The order represents a priority between them and the order is assumed to be total.

Tools written in Prolog were developed which checked the consistency of the security policies as well as an algorithm for solving conflicts when an individual is playing different roles at the same time.

## B. ON THE AXIOMATIZATION OF SECURITY POLICIES: SOME TENTATIVE OBSERVATIONS ABOUT LOGIC REPRESENTATION

In [21], Michael et. al. add an intermediate step to the traditional approach of translating natural language security policies into their axiom representation. Once the policies are in axiom representation, automated reasoning systems are used for the detection of conflicts. Errors in the translation into axiom form can lead to unidentified conflicts, and incorrect proofs when indeed there is a conflict.

An object-oriented approach is introduced to model the security policies using extended entity-relationship (EER) diagrams. The final axioms of the security polices are then derived from the diagrams

[7] A proposition deemed to be self-evident and assumed without proof

rather than directly from the natural language representation. The premise was that overall logic rule formulation is simplified in a model-based approach by capturing many of the rules in the structural model.

A case study comparing the two different approaches, model-based and no pre-structuring, produced results that appear to support a premise that fewer structuring errors are made with the model-based approach. A limitation of the model-based approach is that potential queries which might reveal conflicting security polices may be prevented.

## C. POLICY HIERARCHIES FOR DISTRIBUTED SYSTEMS MANAGEMENT

In [23], Moffett and Sloman form a policy hierarchy is by refining general high-level policies into a number of more specific management policies. This derivation can be performed by refining the goals, partitioning the targets that the policies affect, or delegating the responsibility to another manager. The main motivation for understanding hierarchical relationships between policies is to determine what is required for the satisfaction of policies. If a high-level policy is defined or changed, it should be possible to decide what lower-level policies must be created or changed.

The goal of policy hierarchy analysis is to determine whether:

- The collected lower-level objectives will completely achieve the higher-level objective which they purport to refine.
- There is conflict between the objectives.
- There is a imperatival policy, with a subject, for each objective. An imperatival policy gives an agent the imperative to carry out an action. In most cases this implies obligation.
- There is an authority policy which empowers the subject to achieve the objective. An authority policy provides an agent with the legitimate power to perform an action.

## D. CONFLICTS IN POLICY-BASED DISTRIBUTED SYSTEMS MANAGEMENT

In [18], policies are used as a means to specify the management behavior of a system, without coding the behavior into the manager agents. Lupu and Sloman focus on techniques and tool support for off-line policy conflict detection and resolution. Two types of policies, authorization and obligation, are addressed in this research. Authorization policy specifies what activities a manger is permitted or forbidden to perform on a set of target objects. Obligation policies specify what activities a manager must or must not do to a set of target objects and essentially defines the duties of a manager.

Conflicts can arise in a set of policies, but it is not always desirable to eliminate the conflicts by rewriting the policies or changing the membership of the domains to which policies apply. As automated managers cannot enforce conflicting policies, Lupu and Sloman suggest a precedence relationship must be established between the polices in order to resolve the conflicts. Four types of policy's priority are addressed:

1. Negative policies always have priority : negative policies take precedence over positive ones.
2. The assignment of explicit priorities : policy 1 has priority over policy 2 which has priority over policy 3 ...etc.
3. Distance between a policy and the managed objects : priority is given to the policy applying to the closer class in an inheritance hierarchy. For example, a computer science (CS)

department is a subclass of a university. If a student is in the CS department, policies of the CS department will override those of the university when a conflict exists.

4. Specificity related to domain nesting : a particular case of distance between policies, this principle is that a more specific policy (i.e., a policy applying to a sub-domain) refers to fewer objects so overrides more general policies applying to an ancestor domain.

Lupu and Sloman developed a prototype conflict detection tool that currently detects overlaps between policies and optionally applies domain nesting precedence. The function of the detection tool which is analogous to compile-time type checking for a programming language in that it reduces run-time errors and detects specification errors.

A notation is used to represent policies that is precise and can be analyzed for conflicts using automated tools, but it is not based on a well-known logic. In this system an administrator creates and modifies policies using a policy editor. Checks are made for conflicts, and if necessary policies are modified to remove the conflicts.

Sloman has applied the concept of grouping policies by authorization and obligation, which are then interpreted rather than coded into management agents, in several other works [29, 24, 30].

## E. A FORMAL PROCESS FOR TESTING THE CONSISTENCY OF COMPOSED SECURITY POLICIES

In [22], Michael presents a formal process for testing the logical consistency of composed security policies. The introduction of a structural model is made to represent relationships between security policies, and axiomatizes the policies so that relationships constructed in the model are preserved and made explicit in a logic model. This logic model is then used for deductive proofs of policy consistency. Michael states that problems arise in correctly defining, evaluating, and mapping policies onto procedures and that a structural model reduces these types of gaps.

OTTER, an automated first-order resolution-style theorem prover is used to detect logical contradictions between the axioms in the logic model.

## F. THE PAIR PROJECT: POLICY ANALYSIS OF INTERNET ROUTING

Although not based on formal logic the PAIR tools [26] developed under the PAIR Project [28] provide a means to troubleshoot routing and policy problems in the Internet. These tools are most useful in conjunction with the a Route Server Next Generation (RSng) route server. These route servers can provide a router with it's own view of the network by gathering routing information from neighboring routers, use a route selection procedure, and apply policy requirements for that particular router from the Internet Routing Registry. Once the routing information is processed it is passed using BGP to each router for its own view.

The PAIR tools allow peers to diagnose their routing and policy problems by comparing prescribed policy, i.e. policy registered in the Internet Routing Registry, with policy actually being configured in the Internet. The analysis of routing policy provides the ability to find inconsistencies among policies.

You can also use the PAIR tools to:

- Learn how the policy in the Internet Routing Registry is processed and used to generate Router Server configuration files.
- Troubleshoot global routing problems.
- Identify stale of inaccurate data in the Internet Routing Registry

THIS PAGE INTENTIONALLY LEFT BLANK

## VII. PATH-BASED POLICY LANGUAGE (PPL)

In this section a new language is introduced which is intended to solve and/or alleviate many of those deficiencies which were discussed in the previous sections. This new language, designed by the authors, is called the Path-based Policy Language (PPL)

PPL is designed to support policies that can be applied to both Differentiated Service as well as Integrated Service models proposed by the IETF. Several goals for our new policy language are listed below.

- Create a path-based representation of policies flexible enough to support both path and non-path based traffic flows. For example, providing an absolute path consisting of the links the traffic must take will provide greater control over traffic flows and provide easier support to integrated services. A less specific policy may only need to provide source and destination nodes in its configuration, or perhaps just the specification that all traffic of lets say file transfers, must be forwarded through a specific node acting as a firewall in an edge router.

- Represent network policies in an unambiguous way. This feature will allow us to detect policies that are in conflict as well as create a stable network environment.

- Be abstract enough to cross device and manufacturer boundaries. Providing a language that is too specific will demand constant updates to our language as well as to software on the vendor's devices.

- Have ability to resolve conflicts between policies. In order for conflicts to be resolved, they first have to be detected. Having conflicting policies without the means to detect and resolve them is probably worse than having no policies at all. An obvious example involves security access policies. Having policies defined to restrict access to network resources can build a false sense of security, when a rogue policy conflicts with existing policies to provide access to those same restricted resources. Believing that the restrictive policies are working may prevent network administrators or security personal from verifying the protection, which can have very serious consequences. To support the detection of policy conflicts a follow on process is being developed which will have the ability to translate policies represented in PPL into formal logic. Once the policies are represented in formal logic, a theorem prover can be utilized to detect conflicts.

Figure 8 represents a summary of the constructs of our language. Several examples of possible policies are provided to show the wide range of policies that can be represented in this path-based approach. Using the wild card character of "*", the ability exists to represent explicit paths as well as groups of traffic flows with our language. This flexibility allows us to represent policies based on QoS and at the same time support existing best effort traffic.

In our language, policy conflicts can be resolved using any of three methods:

1) User identification can be used to provide priority of policies based on the creator of that policy.
2) The action items of the policy can be used to assign a priority to an individual policy

3) The action items of the policy can also be used to declare compromises where the priority of the policy is lowered if certain conditions exist. Example 5 shows a policy that similarly lowered its own allocated bandwidth for the general good of the network.

***policyID<userID> @ {paths} {target} {conditions} [{action_items}]***

*policyID*- - unique policy identification token
*userID* - user ID of policy creator
*paths* - network paths the policy affects
*target* - target class of network traffic
*conditions* - any global conditions (items are AND'ed)
*action_items* - for setting parameters (e.g., policy priority), declaring compromises and explicit deny, etc.
*action_item* = [{condition}:] {actions}

Semantics: *policyID* created by *<userID>* dictates that target class of traffic may use *paths* only if *{conditions}* is true after *action_items* are performed.

**Figure 8. Summary of PPL constructs**

The capabilities of our language are illustrated through the use of several examples below. Example 1 shows the ability to specify an explicit path for a traffic flow. Examples 2, 4, 5, 7, and 8 use the "*" to specify partial paths for traffic flows. In examples 3 and 6, the use of "*" places no restrictions on the path the traffic may take.

In example 5 a policy is represented that will make a compromise when certain network conditions are met. This compromise feature provides the ability to throttle back network flows for the general good of the network.

*Example 1:* **Policy 1 <net_manager> @ {<1,2,5>} {class = {faculty}} {*} {priority := 1}**

This is a rule which states that the path starting at node 1, traversing to node 2, and ending at node 5 will provide high priority for faculty users.

*Example 2:* **Policy2 <stone> @ {<*,2,*>, <*,4,*>} {*} time >= 1600, time <= 0800}**

This rule states that all traffic will be allowed to traverse through nodes 2 and 4 during non-working hours. Unless granted by another policy, traffic will not be able to traverse through nodes 2 and 4 during working hours. This is as a result of the default action which is an explicit deny.

*Example 3:* **Policy3 <net_manager> @ {*} {*} {hopCount > 19} {deny}**

This is a rule which states that no path in the network will be permitted if it has a hop count greater than 19. This example shows the ability to use explicit deny.

*Example 4:* **Policy4 <net_manager> @ {<*,5,*> {*} {hostIP = 131.1.*.*}**

All hosts with a network address starting with 131.1 will be permitted to traverse node 5. Having the ability to restrict groups of network addresses as well as individual network addresses is also a part of our language.

*Example 5:* **Policy5 <xie> @ {<1,*,2,*,5>}**
**{traffic_class = {video, voice}, used_bw <= allotted_bw}**
**{allotted_bw = 50M, loss_rate (data) > 40% : allotted_bw := 40M}**

This policy shows the ability for compromise. Voice and Video traffic are provided with an allotted bandwidth of 50 Mb/s, but when the network loss rate is greater than 40%, a compromise will be made to lower the allotted bandwidth to 40 Mb/s. In this example used_bw and allotted_bw are user defined variables. The loss_rate() function is implemented as a message passed from a network device to the policy server.

*Example 6:* **Policy6 <net_manager> @ {*} {traffic_class = {data}} {*} {priority :=10}**

All data traffic will be assigned a priority level of 10. Assume that there are three classes of traffic for this example, voice, video, and data. This allows for providing higher or lower priority to certain classes of traffic. In this case the priority might affect the ordering of packets being dropped from queues in the network during times of congestion.

*Example 7:* **Policy7 <Betty> @ {<1,*,5>} {traffic_class = {accounting}}**
**{day != Friday : priority := 5}**

On all paths from node 1 to node 5, accounting class traffic will be lowered to priority 5 unless it is a Friday. In this policy the action_items field is used with temporal information to influence the priority of a class of traffic. It might make sense to have this feature when departments of a company need more network resources to accomplish their jobs.

*Example 8:* **Policy8 <net_manager> @ {<1,*,5>} {traffic_class = {student}} {*}**
**{userID = Gary : deny}**

On all paths from node 1 to node 5, deny access to network traffic from user Gary who is in the student traffic class. This policy shows that our language can provide control at a very small granularity level. In this case the policy affects only a single user in a particular class of network traffic. It could have easily been modified to provide certain times of the days when it was in effect as well

In this section our new Path-based Policy Language (PPL) was briefly introduced. Our language has the ability to represent network policies unambiguously providing support to heterogeneous networks for which the networks are controlled using explicit policies. Policies required by both path and non-path based traffic flows are supported with our language as well the ability to resolve conflict between policies.

Our current efforts [37] are directed toward fully specifying the grammar of PPL. Examples 1 through 8 in section VII provide examples of our language, but there are support mechanisms that have to be defined to support these policies. Providing the ability for user-defined traffic classes is one example. Constructing a compiler that will verify syntax and detect policy conflicts will take our research forward by providing a mechanism for running experiments.

THIS PAGE INTENTIONALLY LEFT BLANK

# VIII. CONCLUDING REMARKS

In this paper, several policy languages were reviewed to summarize their purpose, strengths, and weaknesses. Two policy-based routing protocols are summarized for their use of policies in networks. In addition, previous work and techniques used in policy conflict resolution and detection using formal logic were also discussed.

These languages fell into two major categories: the abstract network policy languages and bit-level traffic flow languages. The first abstract language is known as the Policy Term, defined by David Clark. It is based on the concept that Internet resources are grouped into Administrative Regions (ARs). The Policy Term was designed to provide for the specification of a wide range of network policies to be represented by supplying source and destination ARs. Other abstract languages are the PFDL (Policy Framework Definition Language), a hierarchical object language being designed by the IETF which is based on the Policy Information Model; and RPSL (Routing Policy Specification Language), the current Internet routing policy specification language. Examples of the languages supporting bit-level details are PDL (Pattern Description Language), used for defining pattern matching criteria in policy-based networking devices; and SRL (Simple Ruleset Language), which specifies the flows to be measured and how much information should be collected for Real-time Traffic Flow Measurement (RTFM).

It is clear that there is a need for policy conflict detection, evident in the fact that many of the languages surveyed have features to resolve conflicts once they are detected. What is lacking is an automatic method for checking network policies that are to be composed together in order to completely satisfy a corporation's policy goals.

One clear method is to use formal logic to represent network policies. Although this method would make conflict detection much easier with the use of existing theorem provers, most network policy implementers are not as comfortable with this representation.

The existing policy languages discussed in this paper are more suited to Differentiated Services rather than Integrated Services as a result of an absence of features to support explicit path-based policies.

Our goal is to develop a network policy language that is more suited toward what network policy implementers are accustomed to: a rule-based representation more closely associated with a computer programming language. Taking a path-based approach will enable us to establish policies that will be based on path, like Integrated Services, as well as non-path based policies which are more suited toward Differentiated Services. The use of a  wild card character enables us to describe policies based on the concepts of Differentiated Services or best-effort traffic.

Path-based policies not only are a natural fit for Integrated Services, but will also aid in the scaling problem which can occur in supporting policies in large networks. A hierarchical view of a network which is provided with concepts such as Autonomous Systems, Administrative Domains, etc. can greatly reduce the information a route server must maintain. One domain can represent multiple hosts which not only reduces the size of the database at a route server, but also the information that must be passed between routing entities. This grouping of nodes into regions is only one aspect in the scalability problem of enforcing policies in networks. The number of policies associated with a domain is a second. Path-based policies can aid in this problem by providing the ability to assign a single policy statement to multiple paths with one statement. For example, the path <1, *, 3> specifies all the possible paths from node 1 to node 3. A policy using this path construct can specify network constraints that are applied to multiple paths. This aggregation of paths can reduce the number of policies statements required at a server.

Path-based policies provide a complexity advantage was well. When a policy server associates policies with nodes rather than paths, a valid path must be constructed for each new request. This construction not only uses node connectivity information to build the possible paths, but applies the policy information from each node as well. If this path generation request involves a combination of service constraints, such as minimum-delay and least-cost, this is now an NP-complete problem. Path-based policy is analogous to the use of static routes. Rather than calculating a route through the network, a valid route is specified ahead of time. This pre-specified route accelerates the routing process. When a path is specified ahead of time with the proper policy constraints, this too will accelerate the response to a path request. PPL provides a compromise to policy representation, by allowing policies to be associated with a node as well as a path.

Our current effort is the development of a compiler to translate policies specified in PPL into formal logic. This will provide us with a means to detect conflicting policies using existing theorem provers. This development will allow us to introduce formal logic into network policy management.

# LIST OF REFERENCES

[1] B. Aiken, J. Strassner, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moore, and B. Teitelbaum, "Terminology for describing middleware for network policy and services," Internet Engineering Task Force Internet Draft draft-aiken-middleware-reqndef-00.txt, April 30, 1999.

[2] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra, "Routing Policy Specification Language (RPSL)," Internet Engineering Task Force Internet Draft draft-ietf-rps-rpsl-v2-03.txt, April 6, 1999.

[3] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust-Management System Version 2," Internet Engineering Task Force: Network Working Group Request for Comments: 2704, September 1999.

[4] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Raja, and A. Sastry, "The COPS (Common Open Policy Service) Protocol", Internet Engineering Task Force, Internet Draft draft-ietf-rap-cops-05.txt, December 1998.

[5] N. Brownleee, "SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups," Internet Engineering Task Force, Internet Draft draft-ietf-rtfm-ruleset-language-07.txt, August 1999

[6] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," Internet Engineering Task Force: Network Working Group Request for Comments: 1157, May 1990

[7] L. Cholvy and F. Cuppens, "Analyzing consistency of security policies," presented at 1997 IEEE Symposium on Security and Privacy, 1997.

[8] D. Clark, "Policy Routing in Internet Protocols," Internet Engineering Task Force: Network Working Group Request for Comments: 1102, May 1989.

[9] A. Guillen, R. N. Kia, and B. Sales, "An architecture for virtual circuit/QoS routing," presented at 1993 International Conference on Network Protocols, 1993.

[10] J. Honig, D. Katz, M. Mathis, Y. Rekhter, and J. Yu, "Application of the Border Gateway Protocol in the Internet," Internet Engineering Task Force: Network Working Group Request for Comments: 1164, June 1990.

[11] C. Kunzinger, "Protocol for the Exchange of Inter-Domain routing Information among Intermediate Systems to Support Forwarding of ISO 8473," Internet Engineering Task Force: working draft ISO 10747, April 1994

[12] J. Kurose and K. Ross, Computer Networking A Top-Down Approach Featuring the Internet, Addison-Wesley, 2000, pp. 152-153.

[13] B. Leiner, "Policy Issues in Interconnecting Networks," Internet Engineering Task Force: Network Working Group Request for Comments: 1124, September 1989.

[14] K. Lougheed and Y. Rekhter, "A Border Gateway Protocol (BGP)," Internet Engineering Task Force: Network Working Group Request for Comments: 1105, May 1989.

[15] K. Lougheed and Y. Rekhter, "A Border Gateway Protocol (BGP)," Internet Engineering Task Force: Network Working Group Request for Comments: 1163, June 1990.

[16] K. Lougheed and Y. Rekhter, "A Border Gateway Protocol 3 (BGP-3)," Internet Engineering Task Force: Network Working Group Request for Comments: 1267, October 1991.

[17] K. Lougheed and Y. Rekhter, "Application of the Border Gateway Protocol in the Internet," Internet Engineering Task Force: Network Working Group Request for Comments: 1268, October 1991.

[18] E. Lupu and M. Sloman, "Conflicts in Policy-based Distributed Systems Management," To appear in IEEE Transactions on Software Engineering - Special Issue on Inconsistency, 1999.

[19]H. Mahon, "Requirements for a Policy Management System," Internet Engineering Task Force, Internet Draft draft-ietf-policy-req-00.txt, September 1999.

[20] D. Meyer, J. Schmitz, C. Orange, M. Prior, and C. Alaettinoglu, "Using RPSL in Practice," Internet Engneering Task Force: Network Working Group Request for Comments: 2650, August 1999.

[21] J. B. Michael, E. H. Sibley, R. Baum, and F. Li, "On the Axiomatization of Security Policy: Some Tentative Observations About Logic Representation," presented at Database Security, VI: Status and Prospects, 1992.

[22] J. B. Michael, "A Formal Process for Testing the Consistency of Composed Security Policies," in Department of Information and Software Systems Engineering. Fairfax: George Mason University, 1993.

[23] J. Moffett and M. Sloman, "Policy Hierarchies for Distributed Systems Management," IEEE Journal on Selected Areas in Communications, vol. 11, pp. 1404-1414, 1993.

[24] J. Moffett, M. Sloman, "User and Mechanism Views of Distributed Systems Management", IEE/IOP/BCS Distributed Systems Engineering Journal, vol 1, no. 1, pp. 37-47, Aug 1993.

[25] M. Nossik, F. Welfeld, and M. Richardson, "PAX PDL – a non-procedural packet description language," http://www.solidum.com/papers/pax-pdel/pax-pdl-00.html, September 30, 1998.

[26] The PAIR Project: Policy Analysis of Internet Routing, http://www.rsng.net/pair/, 1999.

[27] R. Rajan, S. Kamat, J. C. Martin, M. See, R. Chaudhury, D. Verma, G. Powers, and R. Yavatkar, "Policy Action Classes for Differentiated Services and Integrated Services," Internet Engineering Task Force, Internet Draft draft-rajan-policy-qosschema-01.txt, 5 April 1999.

[28] Route Server Next Generation Project, http://www.rsng.net/, 1999.

[29] M. Sloman, "Management Issues for Distributed Services", Proc. IEEE Second International Workshop on Services in Distributed and Networked Environments, pp. 52-59, June 1995.

[30] M. Sloman, "Policy Specification for Programmable Networks", First International Working Conference on Active Networks (IWAN'99), June 1999.

[31] M. Steenstrup, "IDPR: An Approach to Policy Routing in Large Diverse Internetworks", Journal of High Speed Networks, 1994, pp. 81-105.

[32] M. Steenstrup, "An Architecture for Inter-Domain Policy Routing", Internet Engineering Task Force: Network Working Group Request for Comments:1478, June 1993.

[33] M. Steenstrup, "Inter-Domain Policy Routing Protocol Specification: Version 1", Internet Engineering Task Force: Network Working Group Request for Comments:1479, July 1993.

[34] J. Strassner and E. Ellesson, "Terminology for describing network policy and services," Internet Engineering Task Force, Internet Draft draft-strasner-policy-terms-01.txt, 1998.

[35] J. Strassner and S. Schleimer, "Policy Framework Definition Language," Internet Engineering Task Force, Internet Draft draft-ietf-policy-framework-pfdl-00.txt, 17 November 1998.

[36] J. Strassner, E. Ellesson, and B. Moore. (editor), "Policy Framework Core Information Model," Internet Engineering Task Force: Network Working Group, Internet Draft draft-ietf-policy-core-schema-02.txt, February 1999.

[37] G. Stone, "Path-based Policy Language", Naval Postgraduate School, Monterey, CA, Manuscript in preparation, August 2000.

[38] S. Thomas, IPng and the TCP/IP Protocols, Wiley Computer Publishing, 1996, pp. 319-350.

[39] C. Villamizar, C. Alaettinoglu, and D. Meyer, "Routing Policy System Replication," Internet Engineering Task Force, Internet Draft draft-ietf-rps-dist-04.txt, September 28, 1999.

[40] X. Xiao, A. Hanan, B. Bailey, and L.Ni, "Traffic Engineering with MPLS in the Internet," IEEE Network, Vol. 14 No. 2, pp 28-33, March/April 2000

[41] X. Xiao and L. Ni, "Internet QoS: A Big Picture," IEEE Network, Vol. 13 No. 2, pp. 8-18, 1999.

[42] G. G. Xie, D. Hensgen, T. Kidd, and J. Yarger, "SAAM: An integrated network architecture for integrated services," presented at Proceedings of 6th IEEE/IFIP International Workshop on Quality of Service, Napa, CA, 1998.

[43] R. Yavatkar, D. Pendarakis, and R. Guerin, "A Framework for Policy-based Admission Control," Internet Engineering Task Force, Internet Draft draft-ietf-rap-framework-01.txt, November 1998.

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....................................2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virgina 22060-6218

2. Dudley Knox Library.....................................................2
   Naval Postgraduate School
   Monterey, CA 93943-5101

3. Chairman, Code CS ......................................................1
   Naval Postgraduate School
   Monterey, CA 93943

4. Professor Bert Lundy, Code CS ...........................................1
   Naval Postgraduate School
   Monterey, CA 93943

5. Professor Geoff Xie, Code CS ............................................1
   Naval Postgraduate School
   Monterey, CA 93943

6. Communications and Information Systems Department ........................1
   Space and Naval Warfare Systems Center
   Attn: Mr. Mike Harrison
   53560 Hull Street
   San Diego, CA 92152-5001

7. Computational Sciences Division .........................................1
   NASA Ames Research Center
   Attn: Marjori Johnson, Senior Scientist
   MS 269-2 Moffett Field, CA 94035-1000

8. Gary Stone, Code CS .....................................................1
   Naval Postgraduate School
   Monterey, CA 93943

9. Research Office, Code 09 ................................................1
   Naval Postgraduate School
   Monterey, CA 93943